FCRF: Flexible Constructivism Reflection for Long-Horizon Robotic Task Planning with Large Language Models

Yufan Song[1†], Jiatao Zhang[1†], Zeng Gu[2], Qingmiao Liang[2], Tuocheng Hu[1], Wei Song[1⋆], and Shiqiang Zhu[1⋆]

***Abstract*— Autonomous error correction is critical for domestic robots to achieve reliable execution of complex long-horizon tasks. Prior work has explored self-reflection in Large Language Models (LLMs) for task planning error correction; however, existing methods are constrained by inflexible self-reflection mechanisms that limit their effectiveness. Motivated by these limitations and inspired by human cognitive adaptation, we propose the Flexible Constructivism Reflection Framework (FCRF), a novel Mentor-Actor architecture that enables LLMs to perform flexible self-reflection based on task difficulty, while constructively integrating historical valuable experience with failure lessons. We evaluated FCRF on diverse domestic tasks through simulation in AlfWorld and physical deployment in the real-world environment. Experimental results demonstrate that FCRF significantly improves overall performance and self-reflection flexibility in complex long-horizon robotic tasks. Website at https://mongoosesyf.github.io/FCRF.github.io/**

I. INTRODUCTION

Robotic task planning constitutes a fundamental capability for high-level decision-making, enabling robots to generate executable action sequences through environmental perception, capabilities, and task objectives [1]. Domestic robots increasingly assume critical roles in assistive human living, which interact with various objects and generate longer action sequences, often leading to numerous errors that accumulate over time [2]. This characteristic highlights the need for autonomous error correction to ensure stable execution of complex long-horizon task planning.

In recent years, powered by massive data training, rapidly developed LLMs possess encyclopedic world knowledge and situated language comprehension capabilities, enabling their application in robotic task planning [3], [4], [5]. LLMs enhance robotic planning and reasoning capabilities, allowing direct interaction with task instructions in natural language and improving overall task planning performance. This leads us to ask: can an LLM agent with self-reflection capabilities be designed to autonomously correct errors and optimize actions for domestic robots task planning, thereby better addressing the complex needs of long-horizon tasks?

Several existing studies have explored self-reflection mechanisms for error correction in the LLM task planning process. Foundational approaches include RETROFORMER [6], and Reflexion [7] based on ReAct [8] actor, processing reflection gradients through textual representations. Another series of works [9], [10], [11] combine tree search with reflection to deliberately seek a better solution. Recently, works like Expel [12] and AutoManual [13] build rule systems according to the task environment, leading to comprehensive reflection on previous errors of LLM agents.

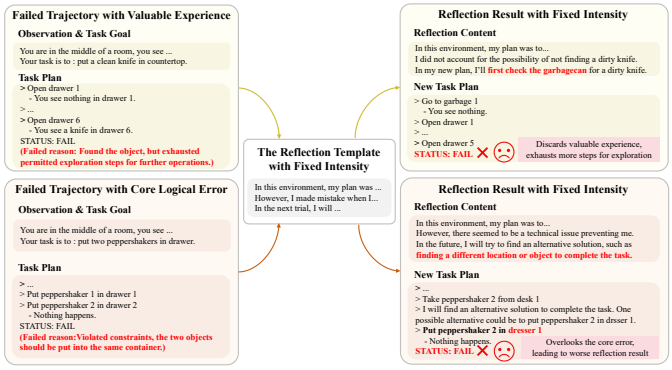

Fig. 1. Illustration of the flexibility problem of self-reflection during long-horizon LLMs task planning. If LLMs always reflect at a fixed intensity, they may discard valuable experience in minor error scence or overlook core errors, leading to suboptimal reflection even worse planning results.

Although previous methods show promising results, a key challenge in applying LLMs for self-reflection to complex long-horizon task planning is their lack of flexibility. Specifically, current frameworks typically employ a fixed reflection paradigm, failing to adjust the intensity of reflection according to the nature of the errors. In long-horizon robotic tasks, the severity of errors varies significantly across different trajectories. For example, some failures result from insufficient exploration, where most actions are correct and only minor adjustments are needed. In such cases, a light reflection suffices. In contrast, failures caused by fundamental logical errors require more intensive reflection and substantial modifications to the trajectory. Therefore, it is crucial for LLMs to integrate deeply with the current task trajectory and adaptively adjust the reflection process. As illustrated in Figure 1, if LLMs reflect at a fixed intensity in complex long-horizon tasks, they risk discarding valuable experience or overlooking the root cause of errors, leading to suboptimal reflection and, ultimately, worse planning results.

Motivated by the challenges mentioned, we investigate the flexibility of LLMs' self-reflection in complex long-horizon robotic task planning. Humans similarly encounter varying degrees of error severity in long-horizon tasks. When errors occur, humans review their actions, analyze useful experience, and integrate external knowledge to refine both successes and failures, reflecting with appropriate intensity to correct the errors. This characteristic extends to widely

[1]Zhejiang University, Hangzhou, China
[2]Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences, Hangzhou, China
⋆Corresponding emails: {weisong-rob,sqzhu}@zju.edu.cn
† Contribute equally to this work.

recognized constructivist learning theory [14], as recognized in educational psychology, underscores that effective learning integrates new and prior experience within the task context. For example, when human students make mistakes, a good educator first acknowledges the correct aspects of their work, then addresses specific errors by drawing on relevant knowledge from a broader system to guide the students. This helps students integrate new and old experience, fostering an appropriately intense reflection.

Inspired by the aforementioned process, we propose the Flexible Constructivism Reflection Framework (FCRF), an online approach enabling LLMs to perform flexible self-reflection based on task difficulty. Our method draws from constructivist learning, using a Mentor-Actor LLM architecture for online reflection. The Actor LLM follows the ReAct [8] framework for action planning, while the Mentor LLM guides the reflection process for error correction. The Mentor summarizes successful experience from the trajectory of Actor and extracts failure lessons from corrected trajectories, maintaining a universal Lesson Pool. Focusing on the issue of flexibility, our framework includes a complexity assessment module, allowing the LLM to select reflection intensity tailored to evaluated task difficulty. Once reflection intensity is determined, valuable experience and failure lessons are constructively integrated into a new plan, guiding the Actor LLM in the planning of next trail.

To evaluate the performance of our framework, we perform experiments in AlfWorld [15] household planning tasks. We divide all the tasks in the dataset into six categories according to their operation type. The results show that our framework achieves the best performance on all tasks, and improves 31.2% reflection flexibility, 25.0% valuable experience recall and 63.3% error correction precision. In summary, the contributions of our work are as follows:

1) To the best of our knowledge, we are the first to define and research the flexibility problem of LLMs reflection for complex long-horizon robotic task planning.
2) We propose a novel Flexible Constructivism Reflection Framework (FCRF), which dynamically determines the intensity of reflection based on task complexity and integrates feedback from task success or failure for adaptive and efficient self-reflection processes.
3) We introduce a Mentor-Actor reflection architecture, where the Mentor leverages a lesson pool, a generalized knowledge base across scenarios. This base may contain new knowledge for human, and supports multiple maintenance methods, including human knowledge injection and LLM-based summarization.
4) We validate our proposed method in both AlfWorld simulation environments and real-world robotic experiments, demonstrating that our framework significantly improves task performance and adaptability under complex and diverse conditions.

II. RELATED WORKS

A. LLMs for Task Planning

Several prior works have employed LLMs for task planning, on account of their inherent powerful reasoning and planning capabilities. For instance, classic LID [16] uses pre-trained GPT-2 [17] as a general framework for interactive decision making, by converting policy inputs including observations, goals, and history into sequential data. These embeddings are then passed to a pre-trained policy network to predict actions. ReAct [8] combines reasoning and acting with language models for solving diverse language reasoning and decision making tasks. Several later works like CodeAsPolicy [18], ProgPrompt [19] and AdaPlanner [20], consider the powerful programming capability of LLMs, propose to use programmatic code as the plan of LLMs. Focusing on the combination of robots and LLMs, typical embodied robotic work SayCan [3] extracts and leverages the knowledge within LLMs in physically-grounded tasks, constraining the model to propose natural language actions that are both feasible and contextually appropriate. A series of other works including LLMPlanner [21], [22] also develop LLMs for use in robotic planning tasks.

B. LLM Agents for Self-Reflection

Developed from simple planning strategies, a series of later studies adopt self-reflection methods, using feedback to perform multistep planning and error correction of LLM agents. For example, improved on the ReAct reasoning framework, Reflexion [7] allows LLMs to reflect on their previous failures according to environmental feedback, forming an improved plan for the next attempt. Based on the Reflexion framework, recent work Expel [12] builds an offline learning process, the LLM agent gathers experience from a collection of training tasks through trial and error. AutoManual [13] builds a well-organized understanding of the environment that can guide multitask planning effectively. Another series of works [9], [10], [11] combine tree search with reflection to seek a better solution to the task.

III. PRELIMINARIES

A. Planning Framework

A task can be defined as a tuple $\langle G, S, O, T, A \rangle$ [23], where G represents the task goal, S represents the set of all possible states, O is the set of observations of task environment, A is the set of possible actions, and T is the transition function, formally formalized as $\mathrm{T} : S \times A \rightarrow S$, represents the environmental state changes because of actions. The objective is to find a plan π, in the form of a sequence of actions, that transitions from the initial state to the target state. There is currently no unified and strict definition of complex long-horizon tasks. Based on the summary of existing methods, we can generally define tasks with more than 10 steps or even longer, simultaneously interacting with a greater variety of items and environments, as complex long-horizon tasks [24].

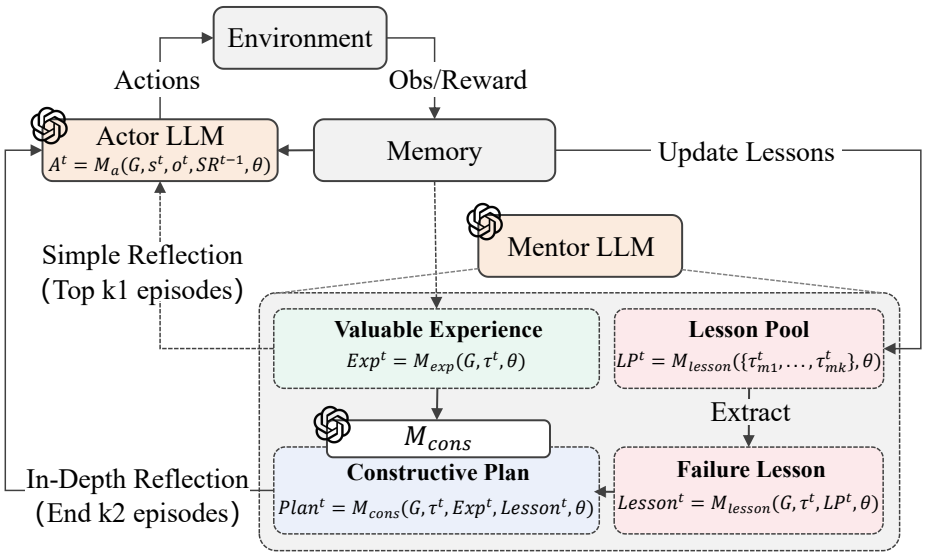

Fig. 2. The framework of FCRF. The planning process is executed by the Actor LLM, the reflection process is executed by the Mentor LLM. During the self-reflection process, the difficulty level of the task is first assessed, according to which the Mentor flexibly selects reflection intensity, determine the proportion of simple experience retain and in-depth failure lessons extraction among all the reflection episodes. Combining the valuable experience and failure lesson, the Mentor performs a constructivism self-reflection to guide the next round of planning of the Actor. The reflection results and planning trajectories will be stored in the memory module for long-term management.

B. Self-Reflection Process in LLMs Planning

Discussing LLMs task planning process with self-reflection and memory module mem, the self-reflection content for current unsuccessful planning trail $sr^t \in SR$ is generated by the LLM for self-reflection M_{sr}, while the reflection process can be described as $M_{sr}(sr^t|s^t, mem^t)$, meaning that the self-reflection content sr^t of the current trail t is generated by M_{sr} based on the current state s^t, and persistent memory mem^t lasts until the current trail including existing task trajectory.

The self-reflection generated by M_{sr} would be stored in the agent memory mem and passed to planner LLM in the next trail, to generate better action sequence $\{a_1^{t+1}, \ldots, a_i^{t+1}\}$ as the new output of the planning task. The sequence of actions is generated by the strategy function $\Phi(a_1^{t+1}, \ldots, a_i^{t+1}|G, s^{t+1}, o^{t+1}, sr^t)$, meaning that new action sequence $\{a_1^{t+1}, \ldots, a_i^{t+1}\}$ from $step_1$ to $step_i$ is generated based on the task goal G, the current agent state s^{t+1}, the current observation of task environment o^{t+1}, and the generated self-reflection content sr^t of last failed trail.

IV. METHODOLOGY

In this section, we present the details of our framework Flexible Constructivism Reflection Framework (FCRF). As illustrated in Figure 2, our FCRF is a Mentor-Actor architecture that consists of four parts: an LLM as planning Actor, an LLM as self-reflection Mentor, the memory management module and the overall flexible reflection process. These modules will be introduced specifically in the following.

A. LLM as Actor

Following the architecture of the classic Reflexion method, our planning process is completed by an LLM prompted as an $Actor$, represented as M_a in the following. M_a takes task goal, current environmental observations and reflection text of previous failed task trails as inputs, and is specifically prompted to generate action sequence in text form as output. This planning process of M_a can be described as:

$$A^t = M_a(G, s^t, o^t, SR^{t-1}, \theta), \tag{1}$$

where $A^t = \{a_1^t, \ldots, a_i^t\}$ denotes the current action sequence, G represents the task goal, o^t represents the current environmental observations during trail t, $SR^{t-1} = \{sr^{t-k}, \ldots, sr^{t-1}\}$ represents the set of self-reflection among the past k trials to consider about, and θ represents the parameters of M_a. The action sequence generated by Actor LLM M_a would be stored in the memory module mem for the following self-reflection process of the task, until the current task is successfully completed.

B. LLM as Constructivism Self-Reflection Mentor

The constructivism self-reflection process of our framework is guided by an LLM prompted as a $Mentor$, represented as M_m, which contains three sub-modules M_{exp}, M_{lesson} and M_{cons}. The reflection process is divided into three parts: the summary of valuable experience, the summary of failure lessons, and the comprehensive construction of the two above parts. The final generated reflection content will be added to the memory module mem and can be called by the Actor LLM M_a in subsequent trails for planning.

The valuable experience summary process is completed by the submodule M_{exp}. M_{exp} analyzes the current trail trajectory planned by M_a, summarizing effective interaction attempts contained in the unsuccessful trail, then retains them as valuable experience and integrates them into the memory,

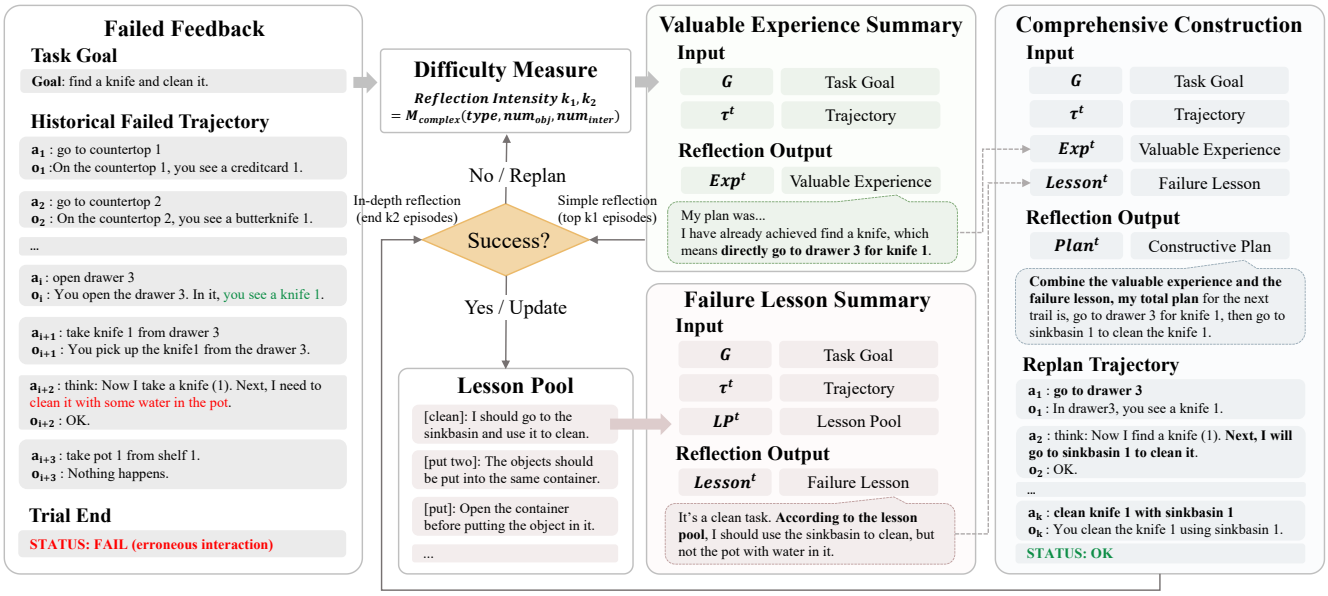

Fig. 3. The detailed illustration of FCRF. The difficulty level of the task is first assessed, determining the number of episodes for simple and in-depth reflection. After the reflection intensity is determined, the detailed self-reflection process consists of the valuable experience summary process, the failure lesson summary process and the comprehensive construction process. Final output of the self-reflection process is a improved new plan for the next attempt.

for usage in the construction of subsequent reflection content. The process can be described as:

$$Exp^t = M_{exp}(G, \tau^t, \theta), \tag{2}$$

where G represents the task goal, τ^t represents the trajectory of current trail t, θ represents the parameters of M_{exp}, mainly embodies through prompt.

The failure lesson summary process is completed by the submodule M_{lesson}, specifically divided into the preliminary maintenance of the mentor lesson pool and the failure lesson extraction process. The mentor lesson pool is expressed as LP, which can be regarded as a universal knowledge base of the task environment. During the lesson pool maintenance process, M_{lesson} accesses the existing cross-task trajectories stored in memory online, summarizing scenario universal failure lessons from all trajectories that have been successfully corrected by the episode in which the current trail locates, then incrementally add the new lesson to LP. The failure lesson summary process can be described as:

$$LP^t = M_{lesson}(\{\tau^t_{m1}, \ldots, \tau^t_{mk}\}, \theta), \tag{3}$$

where LP^t represents the up-to-date lesson pool set up to trail t, $\{\tau^t_{m1}, \ldots, \tau^t_{mk}\}$ represents all k successfully modified task trajectories in the current trail t, θ represents the parameters of M_{lesson}, mainly embodies through prompt.

During the failure lesson extraction process, M_{lesson} combines the current task goal to analyze the current trail task trajectory, then extracts a lesson tip that best fits the cause of the current task failure from the lesson pool maintained so far, as external information for failure reflection. The failure lesson will be used in the construction of subsequent reflection content together with valuable experience Exp^t. The process of failure lesson extraction can be expressed as:

$$Lesson^t = M_{lesson}(G, \tau^t, LP^t, \theta), \tag{4}$$

where G represents the task goal, τ^t represents the trajectory of current trail t, LP^t represents the up-to-date lesson pool set up to trail t, θ represents the parameters of M_{lesson}, mainly embodies through prompt.

After completing the valuable experience summary process and the failure lesson summary process, the constructor submodule M_{cons} will execute **the comprehensive construction process** to comprehensively construct the successful and failed experience, integrate the final reflection result in the form of an improved plan. During the construction process, M_{cons} integrates the summarized valuable experience Exp^t with the newly injected $Lesson^t$, unifying them into a sequence of actions, which forms the new constructive plan to be executed in the next trail. The reflected new constructive plan will be stored in the mem module and guide the planning process of M_{actor} in the next trail. The comprehensive construction process can be described as:

$$Plan^t = M_{cons}(G, \tau^t, Exp^t, Lesson^t, \theta), \tag{5}$$

where G represents the task goal, τ^t represents the trajectory of current trail t, Exp^t and $Lesson^t$ respectively represents the valuable experience and the precisely required failure lesson summarized through trail t, θ represents the parameters of M_{cons}, mainly embodies through prompt.

C. Memory Management

During the task planning process, it is important for LLMs to maintain context memory. However, if all memory contents are included in prompts, it will lead to redundant prompts, increasing the memory burden of LLMs, even affect their performance. Motivated by the long-short-term

memory mechanism of previous work Reflexion, we design the memory module mem for comprehensive memory management, which is equivalent to an external buffer of LLMs. The mem is divided into trajectory management module $Traj^t$ and reflection management module $Refl^t$, comprehensively manages the memory of task trajectories and reflection contents. The $Traj^t$ module is a short-term memory module storing the task trajectory τ^t of the current trail t. The $Refl^t$ module is a long-term memory module storing all failure self-reflection contents up to trail t, which can be represented as $\{sr^1, \ldots, sr^t\}$. The reflection contents are stored for targeted reading and calling by the Mentor LLM M_m in the following trails. Our memory mechanism mem relieves the memory pressure of LLMs, improving the efficiency of memory management and reflection process.

D. The Flexible Reflection Process

As an important innovation, our framework adopts a flexible self-reflection process. For tasks with different difficulties, our method adopts reflection frameworks of different intensities. We design the model $M_{complex}$ to calculate the difficulty level of the task and thus determine the proportion of simple reflection and in-depth reflection. Reflection on simple tasks emphasizes exploration based on valuable experience, while reflection on difficult tasks emphasizes the infusion of external failure lessons. The specific difficulty level assessment process can be expressed as:

$$DL = M_{complex}(type, num_{obj}, num_{inter}), \tag{6}$$

where $type$ represents the type of current task, num_{obj} and num_{inter} respectively represents the number of target objects and the number of interactions that need to be performed within the task. Among all episodes ep_{total}, the number of episodes for in-depth reflection is represented as:

$$k_2 = ep_{total} \cdot \frac{num_{obj} + num_{inter}}{\max\left\{ num_{\text{obj}}(t) + num_{\text{inter}}(t) \,\middle|\, t \in \mathcal{T} \right\}} \tag{7}$$

and the number of episodes for simple reflection $k_1 = ep_{total} - k_2$. The task difficulty assessment and detailed constructivism reflection process are illustrated in Figure 3.

V. EXPERIMENTS

In this section, we evaluate our framework by conducting experiments in common household environment AlfWorld [15]. Furthermore, we perform real-world robotic experiments to verify the practicability of our method in the real world environment. Our approach demonstrates significant advantages in overall performance and specific metrics.

A. Experimental Setup

Environment. We conduct our evaluation on **AlfWorld**, a text-based virtual household environment containing six types (these types would be shown in next part). We test our framework and baselines on entire 134 tasks across all six types in five epochs of planning trails, demonstrating the superiority of our framework in terms of reflection performance and success rate through all methods.

Dataset. In AlfWorld, we conduct experiments on the entire dataset including six types of household tasks: Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place. We perform experiments on all 134 tasks in the environment to obtain results.

Compared methods. We compare our method with three categories of previous methods as main baselines on long-horizon planning tasks: 1. **Planning-Only**: ProgPrompt [19], which LLM receives task descriptions in text form as input, and directly outputs action sequence as planning result based on the In Context Learning (ICL) [25] principle. 2. **Reasoning-Only**: ReAct [8], which combines reasoning and acting processes with LLMs, generating the next action based on reasoning. 3. **Reasoning-Reflection**: Reflexion [7], which allows LLMs to reflect on their previous failures in ReAct trails according to environmental feedback, in order to form an improved plan for the next attempt. The SOTA method Expel [12] is also included in this category of self-reflection methods, while the overall success rate of our method exceeds it on up-to-date GPT-4 series models.

Metrics. We evaluated the methods from three aspects: **Success Rate** wholely evaluates the effectiveness of reflection and planning processes through calculating the proportion of tasks that are completed successfully to the end of trail. **Flexibility** metric includes the Average Value (AVE) and Standard Deviation (STD) of generated self-reflection length, modeling the ability to manage reflection flexibly based on the difficulty of various tasks. **Efficiency** metric includes redefined Experience Recall and Correction Precision during the self-reflection process in terms of action sequence level. The **Experience Recall** metrics the ability to retain valuable experience during the self-reflection process, which can be represented as $Recall_{exp} = \frac{C_{\text{retained}}}{C_{\text{initial}}}$, where $C_{retained}$ denotes the number of correct actions finally retained in the new plan that given by the reflection results, and $C_{initial}$ denotes the number of correct actions initially included in the ultimately failed action sequence. The **Correction Precision** measures the ability to accurately correct wrong steps during the self-reflection process, which can be represented as $Precision_{corr} = \frac{E_{corrected}}{E_{total}}$, where $E_{corrected}$ denotes the number of erroneous actions being corrected through the reflection process, and E_{total} denotes the total number of erroneous actions in the ultimately failed action sequence.

B. Main Results

The main results are presented in Table I and Table II. Balancing performance and computational resource consumption, all experiments are performed using the newest officially recommended *GPT-4o mini* model. We can observe that: 1. Our FCRF outperforms other methods on all tasks and metrics, denoting the ability of our method to enhance the overall success rate of planning, through performing flexible self-reflection according to task difficulty, and constructively integrates experience of both success and failure during the reflection process. An illustrative case of FCRF

TABLE I

SUCCESS RATE OF FCRF AND BASELINES ACROSS VARIOUS ALFWORLD TASKS. OUR FCRF OUTPERFORMS OTHER METHODS ON ALL TASKS.

Methods	Put	Clean	Heat	Cool	Examine	Put two	ALL SR(%)
Planning-Only	64.5	66.6	30.0	62.5	26.6	35.7	68.6
Reasoning-Only	84.6	87.5	77.1	93.3	58.3	88.8	82.8
Reasoning-Reflection	76.9	84.3	80.0	93.3	58.3	96.2	83.5
Ours	**87.5**	**90.0**	**93.5**	**100**	**63.6**	**97.0**	**91.0**

TABLE II

FLEXIBILITY AND EFFICIENCY OF DIFFERENT REFLECTION METHODS. OUR FCRF DEMONSTRATES SIGNIFICANTLY HIGHER FLEXIBILITY AND EFFICIENCY UNDER THE CONDITION OF A 9.7% INCREASE IN COMPUTATIONAL POWER CONSUMPTION.

Methods	Flexibility		Efficiency	
	AVE(words)	STD(words)	Recall$_{exp}$(%)	Precision$_{corr}$(%)
Reasoning-Reflection	371.1	199.9	75.0	32.1
Ours	**407.2**	**262.2**	**100.0**	**95.4**

Fig. 4. Comparison of reflection methods in an AlfWorld example. Faced with the failed trajectory, the Reasoning-Reflection method performs invalid reflection with inappropriate fixed intensity, which discards experience and fabricates a failure reason. While our FCRF first performs simple reflection, summarizing valuable experience to save steps, then FCRF precisely extractes lesson for the error under an in-depth reflection, finally corrects the trajectory.

applied to a long-horizon task is shown in Figure 4, which intuitively demonstrates the superiority of FCRF compared to the baseline method. 2. There is a noticeable trend that weaker self-reflection flexibility and efficiency correspond to an overall lower success rate, which demonstrates the necessity of our research topic and the rationality of metrics we defined. 3. Among all the evaluated methods, the **Planning-Only** method performs simple planning based on inputs and contexts, resulting in a relatively weakest overall success rate. The **Reasoning-Only** and **Reasoning-Reflection** methods achieve better results, with the overall performance of the Reasoning-Reflection method being slightly better than Reasoning-Only. However, in some task categories, Reasoning-Reflection performs worse than Reasoning-Only, demonstrating that self-reflection with fixed templates and strength does not always effectively correct planning errors. 4. Differently from the methods mentioned above, **our FCRF** can flexibly select the intensity of reflection based on the difficulty of the task and constructively integrate the experience of success and failure, thus broadening the overall success rate of varying difficulty tasks, compared to methods of all other categories, as shown in **Table I**. More specifically, as shown in **Table II**, our volume of reflection contents varies more due to the difficulty of task, meanwhile, the average reflection length is only slightly higher than Reasoning-Reflection methods category, demonstrating the **flexibility** and superior cost-performance ratio of our framework. The **efficiency** metric shows that our approach has a higher effective experience recall and core error correction precision, demonstrating the capability of our method in constructing valuable experience and failure lessons, which benefits the overall efficacy of self-reflection.

VI. ANALYSIS AND DISCUSSION

A. Episode Analysis of Self-Reflection Process

We perform episode analysis experiments to specifically analyze the detailed mechanisms and manifestations of the self-reflection process, as shown in Figure 5. We instruct the LLM to continuously replan all erroneous long-horizon tasks over five rounds of experiments, and observe the completion

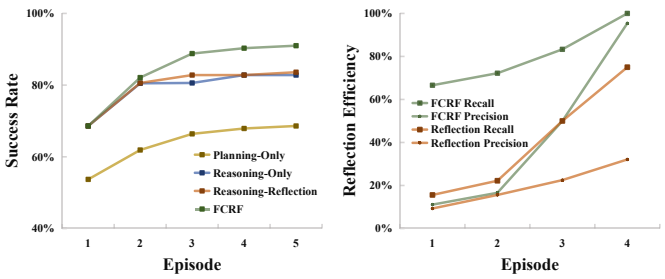

Fig. 5. Result of success rate and reflection efficiency with different episodes. Our FCRF outperforms in all episodes, converges more quickly and has a significant better efficiency.

status on these five episodes. Our observations are as follows. 1. As shown in the left image, our FCRF consistently outperforms the baselines in all episodes and converges more quickly, demonstrating that our method possesses superior performance and robustness together with a better computational cost efficiency. 2. In all episodes, the simple Planning-Only method exhibits the weakest error correction performance, the Reasoning-Only and Reasoning-Reflection methods achieve performance improvements, but there is no noticeable gap between the two methods. 3. As shown in the right image, our FCRF has a significantly stronger ability to recall valuable experience in the trajectory. After the lesson pool has been constructed online, our FCRF demonstrates significantly higher precision in error correction. This further illustrates that reflection process with fixed intensity is not always necessarily conducive to error correction, while our implementation of reflection flexibility and the integration of valuable experience together with failure lessons are indeed conducive to the self-reflection process.

B. Ablation Study

TABLE III

ABLATION OF MODULES OF FCRF IN ALFWORLD TASKS. ALL DESIGNED MODULES CONTRIBUTE TO THE PERFORMANCE.

	SR	AVE	STD	Recall$_{exp}$(%)	Precision$_{corr}$(%)
Ours full	91.0	407.2	262.2	100.0	95.4
w/o Experience	87.3	238.4	82.8	37.5	88.9
w/o Lesson	88.8	276.2	110.7	100.0	20.0

To fully demonstrate the effectiveness of each module in our framework and to explore the interrelationships between them, we perform ablation studies on the full AlfWorld dataset, and the results are detailed in Table III. In the **w/o Experience** model, the action sequence is planned without extracting valuable experience. Compared to the complete model, the overall success rate decreases by 3.7%. The ability to recall valuable experience obviously reduces to 37.5%, while the ability to precisely correct errors still exists. In the **w/o Lesson** model, the action sequence is planned without using failure lesson, the success rate decreases by 2.2%. The ability to precisely correct errors reduces to 20% while the ability to recall the experience is still great. The flexibility of the method decreases when each module is ablated, leading to a decrease of overall success rate, but the performance of each ablated model is still better than other baselines. Altogether, results of the ablation study met expectations on each metric, demonstrating the effect of each module in our method, and showing that our framework the whole is greater than the simple sum of its parts.

C. The Extracted Lesson Pool Result

In our Mentor-Actor reflection architecture, as mentioned in methodology section, the Mentor LLM visits successfully corrected trajectories of the Actor LLM through an online process, extracting a lesson pool to guide the subsequent reflection. The automatically extracted lesson pool is equivalent to a generalized knowledge base in task scenarios, which may contain environmental constraints that humanity has not yet recognized, thus possesses the significance of retention and generalization. The lesson pool obtained during our experimental process is partly displayed in Figure 6.

Automatically Extracted Lesson Pool

[puttwo]: When tasked with finding multiple items, prioritize locating and placing each item **in the same designated container** sequentially.

[clean]: To successfully complete a task involving cleaning an object, I should first locate the object, then clean it directly at the appropriate location (**usually like a sink**).

[put]: When putting an object in a container, I should first find and take the object, then **open the container** before putting the object in it.

[heat]: Systematically check all potential locations for the required items, ensuring to interact with each object correctly before proceeding to the next step, **go to a microwave and heat with it**.

...

Fig. 6. Part of the automatically extracted lesson pool by the Mentor LLM in our experiment. The lesson pool may contain environmental constraints with the significance of retention and generalization.

D. Real-World Robotic Experiment

We use a quadruped robot with a manipulator to validate the practicability of our method in the real world. The results indicate that the robot deployed with our FCRF performs better in self-reflection on a block organization task. The detailed process is shown in Figure 7. More details of real-world experiments will be shown in our website and video.

VII. CONCLUSIONS

To our knowledge, we are the first to study the problem of self-reflection flexibility and constructivism in long-horizon robotic task planning with LLMs. Based on the constructivist learning theory of human intelligence, we propose a Mentor-Actor self-reflection framework called FCRF, which performs self-reflection with intensity flexibility according to task difficulty, meanwhile integrating valuable experience and failure lessons. Furthermore, we design a memory management module to efficiently manage the reflection context of LLMs. Experiments conducted in virtual household environment and the real world demonstrate that our reflection framework can effectively enhance the flexibility, efficiency and success rate of long-horizon planning tasks, making domestic robots more reliable and trustworthy.

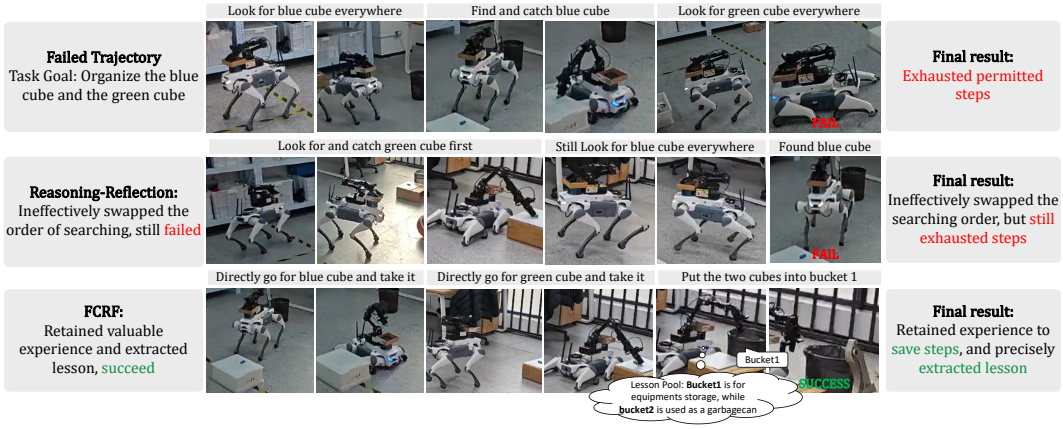

Fig. 7. Comparison of reflection methods in a real-world experiment. In the block organization task, Reasoning-Reflection invalidly swaps the order of searching, while FCRF retains experience of object position to save action steps and extracts precise lesson for constraints, finally corrected the trajectory.

REFERENCES

[1] Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. Recent trends in task and motion planning for robotics: A survey. *ACM Computing Surveys*, 55(13s):1–36, 2023.

[2] Georgios A Zachiotis, George Andrikopoulos, Randy Gornez, Keisuke Nakamura, and George Nikolakopoulos. A survey on the application trends of home service robotics. In *2018 IEEE international conference on Robotics and Biomimetics (ROBIO)*, pages 1999–2006. IEEE, 2018.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[4] Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12140–12147. IEEE, 2024.

[5] Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2081–2088. IEEE, 2024.

[6] Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*, 2023.

[7] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[8] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[10] Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*, 2024.

[11] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

[12] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

[13] Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Generating instruction manuals by llm agents via interactive environmental learning. *arXiv preprint arXiv:2405.16247*, 2024.

[14] George E Hein. Constructivist learning theory. *Institute for Inquiry. Available at:/http://www. exploratorium. edu/ifi/resources/constructivistlearning. htmlS*, 1991.

[15] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[16] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022.

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[18] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[19] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[20] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[21] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

[22] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application. *IEEE Access*, 2023.

[23] Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

[24] Jiatao Zhang, Lanling Tang, Yufan Song, Qiwei Meng, Haofu Qian, Jun Shao, Wei Song, Shiqiang Zhu, and Jason Gu. Fltrnn: Faithful long-horizon task planning for robotics with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6680–6686. IEEE, 2024.

[25] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.